

# CHString(Format)

Be careful with string formatting operations

Sean Barnum, Digital, Inc. [vita<sup>1</sup>]

Copyright © 2007 Digital, Inc.

2007-03-19

## Part "Original Digital Coding Rule in XML"

Mime-type: text/xml, size: 5220 bytes

<b>Attack Category</b>	<ul style="list-style-type: none"><li>• Malicious Input</li></ul>								
<b>Vulnerability Category</b>	<ul style="list-style-type: none"><li>• Format string</li><li>• Unconditional</li></ul>								
<b>Software Context</b>	<ul style="list-style-type: none"><li>• String Formatting</li></ul>								
<b>Location</b>	<ul style="list-style-type: none"><li>• CHString (MFC)</li></ul>								
<b>Description</b>	<p>Incautious use of CHString formatting operations can open vulnerabilities to format attacks.</p> <p>The CHString class has a variety of string manipulation methods. Some methods accept a format string and operate like sprintf to format the remaining variable arguments into a message string. Functions that accept format strings are typically vulnerable to format string attacks.</p> <p>Note: CHString is part of the WMI Provider Framework classes. The Provider Framework is obsolete and not recommended. See Using WMI (referenced below) for the preferred ways to write a WMI COM provider or a WMI provider that uses the .NET Framework System.Management namespaces.</p>								
<b>APIs</b>	<table border="1"><thead><tr><th>FunctionName</th><th>Comments</th></tr></thead><tbody><tr><td>CHString::Format</td><td>Src: 1 variable; Fmt: 0;</td></tr><tr><td>CHString::FormatMessage</td><td>Src: 1 variable; Fmt: 0;</td></tr><tr><td>CHString::FormatV</td><td>Src: 1 variable; Fmt: 0;</td></tr></tbody></table>	FunctionName	Comments	CHString::Format	Src: 1 variable; Fmt: 0;	CHString::FormatMessage	Src: 1 variable; Fmt: 0;	CHString::FormatV	Src: 1 variable; Fmt: 0;
FunctionName	Comments								
CHString::Format	Src: 1 variable; Fmt: 0;								
CHString::FormatMessage	Src: 1 variable; Fmt: 0;								
CHString::FormatV	Src: 1 variable; Fmt: 0;								
<b>Method of Attack</b>	<p>If an attacker can control formatting text, a Format string attack could be mounted.</p> <p>These functions can be abused in much the same way sprintf() can be. If the attacker embeds a series of "%s" format fields in the data, the function will interpret them as formatting commands. It will get the corresponding data values off the stack. With enough "%s" fields, the function will eventually dereference a bad pointer (e.g. NULL) and crash. If</p>								

1. <http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html> (Barnum, Sean)

	the attacker includes "%n" fields, the function will write values to memory, which may ultimately be exploitable.						
<b>Exception Criteria</b>							
<b>Solutions</b>	<table border="1"> <thead> <tr> <th><b>Solution Applicability</b></th><th><b>Solution Description</b></th><th><b>Solution Efficacy</b></th></tr> </thead> <tbody> <tr> <td>Generally applicable to CHString format methods.</td><td>Do not use a format string that may come from or have been tampered with by an untrustworthy source.</td><td>Effective.</td></tr> </tbody> </table>	<b>Solution Applicability</b>	<b>Solution Description</b>	<b>Solution Efficacy</b>	Generally applicable to CHString format methods.	Do not use a format string that may come from or have been tampered with by an untrustworthy source.	Effective.
<b>Solution Applicability</b>	<b>Solution Description</b>	<b>Solution Efficacy</b>					
Generally applicable to CHString format methods.	Do not use a format string that may come from or have been tampered with by an untrustworthy source.	Effective.					
<b>Signature Details</b>	<pre>CHString::Format(LPCWSTR); CHString::Format(UINT); CHString::FormatMessageW(LPCWSTR); CHString::FormatMessageW(UINT); void CHString::FormatV( LPCWSTR lpszFormat, va_list argList );</pre>						
<b>Examples of Incorrect Code</b>	<pre>CHString aString; aString.Format(userSuppliedText);</pre>						
<b>Examples of Corrected Code</b>	<pre>CHString aString; aString.Format( "%s" , userSuppliedText );</pre>						
<b>Source Reference</b>	<ul style="list-style-type: none"> <li>• <a href="http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/vclrfcstringclassmembers.asp">http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/vclrfcstringclassmembers.asp</a><sup>2</sup></li> </ul>						
<b>Recommended Resources</b>	<ul style="list-style-type: none"> <li>• MSDN reference for CHString<sup>3</sup></li> <li>• MSDN reference for CStringT<sup>4</sup></li> <li>• MSDN reference for FormatMessage, suitable for debugging (uses Src: 7 variable; Fmt: 5)<sup>5</sup></li> <li>• MSDN: Using WMI<sup>6</sup></li> </ul>						
<b>Discriminant Set</b>	<table border="1"> <tr> <td><b>Operating System</b></td><td>• Windows</td></tr> <tr> <td><b>Languages</b></td><td>• C • C++</td></tr> </table>	<b>Operating System</b>	• Windows	<b>Languages</b>	• C • C++		
<b>Operating System</b>	• Windows						
<b>Languages</b>	• C • C++						

## Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Digital, including information about “Fair Use,” contact Digital at [copyright@digital.com](mailto:copyright@digital.com)<sup>1</sup>.

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

---

1. <mailto:copyright@digital.com>